An Artificial Fish Swarm Filter-Based Method for Constrained Global Optimization

Ana Maria A.C. Rocha[1,4], M. Fernanda P. Costa[2,3] and Edite M.G.P. Fernandes[4]

[1] Department of Production and Systems, School of Engineering,
arocha@dps.uminho.pt
[2] Department of Mathematics and Applications, School of Sciences,
[3] Mathematics R&D Centre,
mfc@math.uminho.pt
[4] Algoritmi R&D Centre,
emgpf@dps.uminho.pt
University of Minho, 4710-057 Braga, Portugal

Abstract. An artificial fish swarm algorithm based on a filter methodology for trial solutions acceptance is analyzed for general constrained global optimization problems. The new method uses the filter set concept to accept, at each iteration, a population of trial solutions whenever they improve constraint violation or objective function, relative to the current solutions. The preliminary numerical experiments with a well-known benchmark set of engineering design problems show the effectiveness of the proposed method.

Keywords: Global optimization; Swarm intelligence; Artificial Fish Swarm; Filter Method

1 Introduction

In this paper, a new method for handling general nonlinear constraints in a global optimization problem is proposed. The method is based on the implementation of a filter methodology within a population-based swarm intelligence algorithm for solving continuous nonlinear constrained global optimization problems in the form:

$$\begin{array}{l} \underset{x\in\Omega}{\text{minimize}} \quad f(x) \\ \text{subject to } g_j(x) \leq 0\,, j = 1, \ldots, p \end{array} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $g_j : \mathbb{R}^n \to \mathbb{R}$ are nonlinear continuous functions and $\Omega = \{x \in \mathbb{R}^n : -\infty < l_k \leq x_k \leq u_k < +\infty, k = 1, \ldots, n\}$. Problems with equality constraints can be reformulated in the above form using a small tolerance. This is a common procedure in stochastic methods for global optimization. Since we do not assume convexity, problem (1) may have several minima and convergence to the global minimum is not guaranteed by some classical gradient-based algorithms. Derivative-free deterministic and stochastic methods are available to

solve global optimization problems, in particular bound constrained problems. Algorithms that use nature inspired or swarm intelligence principles are common in the literature, see for example [5, 14, 16, 21–23, 30, 35, 36, 42]. A recent artificial life computing algorithm that simulates fish swarm behaviors has been used in different contexts [18–20, 39, 41]. The algorithm known as the artificial fish swarm (AFS) algorithm has shown to be competitive with other global solution methods [30].

Most stochastic as well as deterministic methods for global optimization were firstly developed for unconstrained or simple bound constrained problems. Then, they were extended to more general constrained problems by modifying the solution procedures or by applying penalty function methods. Constraint-handling techniques for global optimization can be classified according to the below referred categories.

- Methods based on penalty functions, where the constraint violation is combined with the objective function to define the penalty function that aims at penalizing infeasible solutions [7, 10, 24, 27, 34]. Augmented Lagrangian techniques are particular cases of penalty methods [8, 31, 32].
- Methods based on multi-objective optimization concepts, where both constraint violation and objective function are goals to be minimized separately [1, 3, 17]. The dominance concept of optimality in the multi-objective optimization field is used to accept trial solutions.
- Methods based on biasing feasible over infeasible solutions, where constraint violation and objective function are used separately and optimized by some sort of order being the violation the most important [6, 13, 33, 40, 42].
- Methods that give superiority to feasible solutions, in which feasible solutions are always better than the infeasible ones [13, 22, 29].
- Methods based on preserving feasibility of solutions, where infeasible points are discarded or repaired [9, 42].
- Methods that use an ensemble of (selected) constraint handling techniques, where each technique has its own subpopulation and it is chosen according to the characteristics of the problem to be solved and the stage of the iterative process [26].

Although penalty function methods are probably the most known constraint handling technique, a penalty function depends, in general, on a penalty parameter. Unfortunately, the choice of a suitable value for the penalty parameter is a critical issue since it depends on the optimal solution of the problem. Fletcher and Leyffer [15] proposed a filter method as an alternative to penalty functions to guarantee global convergence in algorithms for nonlinear optimization. This technique incorporates the concept of non-dominance from the multi-objective programming to build a filter that is able to accept solutions if they improve either the objective function or the constraint violation, instead of a linear combination of those two measures. Therefore, the filter replaces the use of penalty functions, avoiding the update of their penalty parameters. The filter methodology has already been used with sequential quadratic programming and interior point methods for solving nonlinear optimization problems, see for example [11,

12, 15, 28, 37, 38]. Convergence to a local optimal solution, although not necessarily a global one, has been guaranteed whatever the initial approximation. A derivative-free pattern search filter method for nonlinear constrained optimization has already been proposed [4]. However, the therein convergence analysis requires specific problem structure. In the field of global optimization, Hedar and Fukushima present in [17] a hybrid simulated annealing method that uses a filter set concept to accept trial solutions exploring both feasible and infeasible regions.

In this paper, we are particularly interested in using the filter methodology within the AFS algorithm to efficiently solve constrained global optimization problems. The algorithm does not compute or approximate any derivatives or penalty parameters, and will be hereafter denoted by AFSFilter. The method uses the filter set concept to accept, at each iteration, a population of trial solutions whenever they improve constraint violation or objective function relative to the current solutions. This is the first attempt to incorporate the filter methodology into a population-based algorithm to handle the constraints of the problem.

In nature, fishes desire to stay close to the swarm, protecting themselves from predators and looking for food, and to avoid collisions within the group. These behaviors inspire mathematical modelers aiming to solve efficiently optimization problems. The main fish swarm behaviors are the following:

i) *random* behavior - in general, fish swims randomly in water looking for food and other companions;
ii) *searching* behavior - this is a basic biological behavior since fish tends to the food; when fish discovers a region with more food, by vision or sense, it goes directly and quickly to that region;
iii) *swarming* behavior - when swimming, fish naturally assembles in groups which is a living habit in order to guarantee the existence of the swarm and avoid dangers;
iv) *chasing* behavior - when a fish, or a group of fishes, in the swarm discovers food, the others in the neighborhood find the food dangling quickly after it.

The artificial fish is a fictitious entity of a true fish. Its movements are simulations and interpretations of the above listed fish behaviors [20, 39]. The environment in which the artificial fish moves, searching for the minimum, is the feasible search space of the minimization problem. Considering the problem that is addressed in the paper, the feasible search space is the set $\Xi = \{x \in \mathbb{R}^n : g_j(x) \leq 0\,, j = 1, \ldots, p \text{ and } l_k \leq x_k \leq u_k\,, i = k, \ldots, n\}$. The position of an artificial fish in the solution space is herein denoted by a point x (a vector in $\mathbb{R}^n$). We will use the words 'fish' and 'point' interchangeably throughout the paper.

The organization of the paper is as follows. In Section 2, the filter paradigm is briefly introduced. Section 3 describes the AFS algorithm and presents the details concerning with the use of the filter methodology within the AFS algorithm to handle the constraints of the problem. Section 4 describes the numerical experiments of this preliminary study and Section 5 presents the conclusions and ideas for future work.

Notation: $x^i \in \mathbb{R}^n$ is used to represent the ith point of a population, $x^i_k \in \mathbb{R}$ is the kth ($k = 1, \ldots, n$) component of the point x^i of the population, and p_{size} is the number of points in the population. x^{best} is the best point of the population in the sense that it is better than any other point in the population (see Definition 1). The objective function value of the best point of the population is denoted by f^{best}, and x^* is the global optimal solution. The iteration counter in the algorithm is t and $t_{\max}$ represents the maximum number of allowed iterations. $\|.\|$ represents the Euclidean norm.

2 Filter Paradigm

The filter paradigm aims at accepting trial solutions of optimization problems if they improve either the constraint violation or the objective function, with respect to current solutions. The methodology appears naturally from the observation that an optimal solution of a nonlinear optimization problem like (1) aims at minimizing both constraint violation and objective function values,

$$\theta(x) \doteq \sum_{j=1}^{p} \max\{0, g_j(x)\} \quad \text{and} \quad f(x), \tag{2}$$

respectively. Thus, problem (1) is seen as a bi-objective problem, with two goals, where θ is the objective with the highest priority because we must ensure that $\theta(x^*) = 0$. The methodology uses the concept of non-dominance borrowed from the multi-objective optimization. In this context, a point x^i, or the corresponding pair $(\theta(x^i), f(x^i))$, is dominated by a point x^j, or the corresponding pair $(\theta(x^j), f(x^j))$, if

$$\theta(x^j) \leq \theta(x^i) \text{ and } f(x^j) \leq f(x^i).$$

Clearly, a trial solution is considered better than the current solution if it is not dominated by the current solution. The filter $\mathcal{F}$ is defined as a finite set of pairs $(\theta(x^j), f(x^j))$ that correspond to a collection of infeasible solutions x^j such that no filter entry is dominated by any of the others in the filter. The filter defines a forbidden region that do not accept solutions that are dominated by pairs in the current filter. Only solutions that are not dominated by any pair in the filter might be accepted. To avoid acceptance of a trial solution that corresponds to a pair that is arbitrarily close to the border of the filter, the acceptability condition of a solution x to the filter is:

$$\theta(x) \leq (1 - \gamma_\theta)\, \theta(x^j) \text{ or } f(x) \leq f(x^j) - \gamma_f\, \theta(x^j) \tag{3}$$

for all points x^j in the current filter, i.e., for all filter entries $(\theta(x^j), f(x^j)) \in \mathcal{F}$, where $\gamma_\theta, \gamma_f \in (0, 1)$. A typical filter is shown in Figure 1. The shaded area represents the region dominated by the filter entries. According to the conditions in (3), all pairs that are below and to the left of the dashed line are acceptable to the filter. When a solution x is added to/included into the filter, then all the entries that are dominated by the new entry $(\theta(x), f(x))$ are removed from the

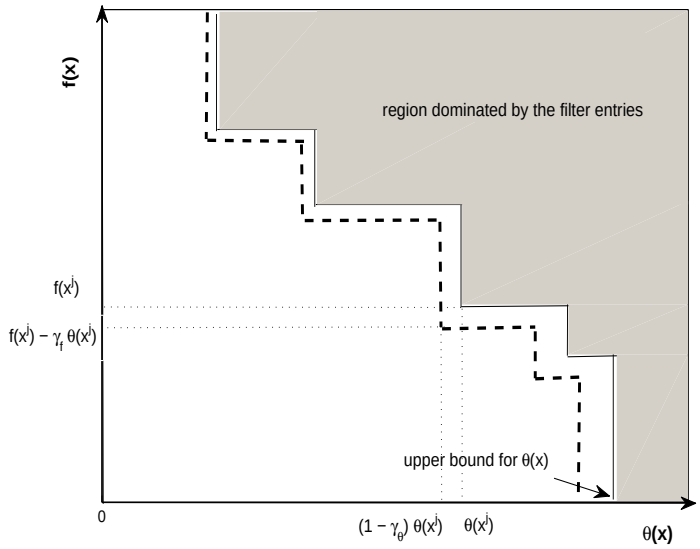

Fig. 1. A filter with four entries.

filter. We remark that an inclusion into the filter can occur only when $\theta(x) > 0$. Figure 2 shows that $(\theta(x^i), f(x^i))$ is removed since it is dominated by the new entry $(\theta(x^j), f(x^j))$.

3 The AFSFilter Method

The details concerned with the procedures of the AFS algorithm as well as the implementation of the filter methodology in the proposed population-based method are now described. The AFSFilter method uses:

i) the artificial fish swarm algorithm to define, at each iteration, random movements and a set of trial solutions;
ii) the filter methodology to define the acceptability conditions that are able to accept trial solutions according to their constraint violation and objective function values.

Progress towards the optimal solution is assessed by the filter methodology. Here, a global optimal solution $x^* \in \Omega$ such that

$$g_j(x^*) \leq 0\,, j = 1, \ldots, p \text{ and } f(x^*) \leq f(x) \text{ for all } x \neq x^* \in \Omega$$

is to be found. In this context, when comparing points of the population, the following definition is used:

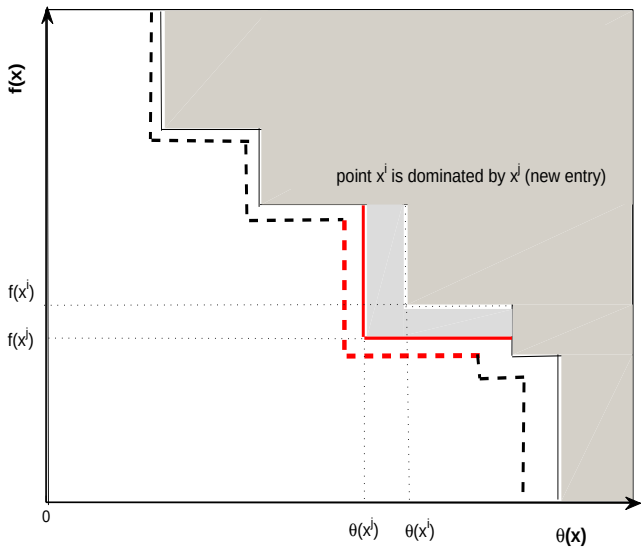

Fig. 2. The entry $(\theta(x^j), f(x^j))$ removes $(\theta(x^i), f(x^i))$.

Definition 1. *Let x^i and x^j be two points inside Ω. x^i is a better point than x^j if the following condition holds:*

$$\theta(x^i) < \theta(x^j) \;\; or \;\; \left(\theta(x^i) = \theta(x^j) \; and \; f(x^i) < f(x^j)\right). \tag{4}$$

To define an appropriate movement for a point x^i in the population, the AFS algorithm relies on a crucial quantity: the 'visual scope' of the point. This represents the closed neighborhood with center x^i and radius equal to a positive quantity v. Based on the simple bounds of the variables, in the problem (1), v is defined by

$$v = \varsigma \max_{k \in \{1,\ldots,n\}} (u_k - l_k),$$

where ς is a positive visual parameter. Relative to x^i , let

- I^i be the set of indices of the points inside the 'visual scope' ($i \notin \mathrm{I}^i$),
- n_{p}^i be the number of points in its 'visual scope' ($n_{\mathrm{p}}^i < p_{\mathrm{size}}$).

If the condition $n_{\mathrm{p}}^i / p_{\mathrm{size}} \leq \kappa$ holds, where $\kappa \in (0, 1]$ is the crowd parameter, the 'visual scope' of x^i is said to be not crowded. Depending on the relative positions of the points in the population, the below three possible situations may occur.

1. When $n_{\mathrm{p}}^i = 0$, the 'visual scope' is empty, and the point x^i, with no other points in its neighborhood to follow, has a *random* behavior.

2. When the 'visual scope' is crowded, the point has some difficulty in following any particular point, and has a *searching* behavior. It chooses randomly another point from the 'visual scope', hereafter denoted by x^{rand}, and moves towards it if x^{rand} is better than x^i (see condition (4)); otherwise it moves according to a *random* behavior.
3. When the 'visual scope' is not crowded, the point firstly tries the *chasing* behavior moving towards the best point inside the 'visual scope', denoted by $x^{\min}$, if this is better than x^i. Otherwise, the point first tries the *swarming* behavior moving towards the central point, c, of the 'visual scope'. However, if c is not better than x^i, the point follows either a *searching* behavior or a *random* behavior depending on the point x^{rand} being better than x^i or not.

We remark that the described procedures carried out when the 'visual scope' is not crowded correspond to a modification that has been introduced in the original version of the AFS algorithm ([20, 39]) and has been shown to outperform the original algorithm, see for example [30, 32].

A simple formal description of the AFSFilter algorithm is presented below in Algorithm 1. The algorithm terminates with a successful message if the best solution found thus far is feasible and is within a certain percentage of accuracy of the best known optimal solution found in the literature, f^{opt}. The algorithm is also allowed to run for a maximum of $t_{\max}$ iterations.

Details related with the numerical computations to define point movements and translate the previously referred behaviors will be presented below. We remark that these movements have been devised to maintain the points satisfying the bound constraints of the problem, i.e., inside the set Ω.

Firstly, the initial population is randomly generated in the set Ω. Each point x^i in the population is componentwise computed by

$$x_k^i = l_k + \lambda(u_k - l_k), \text{ for } k = 1, \ldots, n,$$

where u_k and l_k are the upper and lower bounds respectively of the set Ω, and λ is an independent uniform random number distributed in the range $[0, 1]$.

For each point x^i of the current population, the trial point y^i is generated according to a direction d and a step size $\alpha \in (0, 1]$,

$$y^i = x^i + \alpha d. \tag{5}$$

The procedure that decides if a trial solution is to be accepted and replaces the current solution is a filter method, as explained later on in Subsection 3.5.

3.1 *Random* Behavior

When a *random* behavior is invoked, the point x^i moves randomly and

$$y^i = x^i + \alpha\, \delta\, RNG,$$

where δ is a uniformly distributed number between -1 and 1 and RNG is a vector whose components denote the allowed range of movement towards the lower bound l_k, or the upper bound u_k, of the set Ω, for each component k.

Algorithm 1 AFSFilter algorithm

Require: Random population $x^i \in \Omega$, for $i = 1, \ldots, p_{\text{size}}$, $t_{\max} > 0$, $0 < \epsilon_1, \epsilon_2 \ll 1$

Set $t = 0$
Compute f and θ for all x^i and select x^{best}
while $\left(\left|f^{\text{best}} - f^{\text{opt}}\right| > \epsilon_1 \left|f^{\text{opt}}\right| + \epsilon_2 \quad \text{or} \quad \theta^{\text{best}} > \epsilon_2\right) \quad \text{and} \quad t \leq t_{\max}$ **do**
  for all x^i **do**
    Compute 'visual scope'
    if 'visual scope' is empty **then**
      Random behavior
    else
      if 'visual scope' is crowded **then**
        Select randomly x^{rand} from the 'visual scope'
        if x^{rand} is better than x^i **then**
          Searching behavior
        else
          Random behavior
        end if
      else
        if $x^{\min}$ is better than x^i **then**
          Chasing behavior
        else
          Compute c
          if c is better than x^i **then**
            Swarming behavior
          else
            Select randomly x^{rand} from the 'visual scope'
            if x^{rand} is better than x^i **then**
              Searching behavior
            else
              Random behavior
            end if
          end if
        end if
      end if
    end if
    Filter line search to decide if the trial point y^i is accepted
  end for
  Set $t = t + 1$
end while

3.2 *Searching* Behavior

When the 'visual scope' is crowded, the AFSFilter algorithm tries to follow the *searching* behavior. A point inside the 'visual scope' is randomly selected, x^{rand} (rand $\in \mathrm{I}^i$), and the point x^i is moved towards it to generate a trial point y^i if x^{rand} is better than x^i (see (4)). Here, the direction of movement to get y^i is defined as $d = x^{\text{rand}} - x^i$, recall (5). Otherwise, the point x^i follows a *random* behavior as previously described in Subsection 3.1.

3.3 *Chasing* Behavior

According to Algorithm 1, *chasing* behavior is performed if the 'visual scope' is not crowded and the best point inside the 'visual scope' of x^i, $x^{\min}$, is better than x^i, in the sense of Definition 1. The direction to carry out the movement towards a trial point y^i is defined by $d = x^{\min} - x^i$.

3.4 *Swarming* Behavior

When the 'visual scope' of a point x^i is not crowded and $x^{\min}$ is not better than x^i, the central point inside the 'visual scope' of x^i is computed by

$$c = \sum_{j \in \mathrm{I}^i} x^j / n_{\mathrm{p}}^i \,,$$

and compared with x^i. If c is better than x^i, then the *swarming* behavior follows and the corresponding trial point is computed using (5) where the direction of the movement is $d = c - x^i$. Otherwise, the *searching* behavior is tried (see Subsection 3.2).

3.5 The *Filter Line Search* Method

Here, we aim to show how the methodology of a filter as outlined in [15] can be adapted to this population-based AFS method. Each entry in the filter is defined by two components: $\theta(x)$ that aims to measure feasibility and $f(x)$ that measures optimality, as previously defined in (2). The proposed paradigm defines just one filter. After a search direction d has been computed, a step size should be determined, using (5), in a way that sufficient progress towards the optimal solution is obtained.

Backtracking Line Search The step size $\alpha \in (0, 1]$ is determined by a backtracking line search technique. A decreasing sequence of step sizes $\{\alpha_j\}$ with $\lim_j \alpha_j = 0$ is tried, until a set of acceptance conditions are satisfied. This j denotes the iteration counter for the inner loop. A trial step size α_j can be accepted if the corresponding trial point $y^i = x^i + \alpha_j d$ is acceptable by the filter. When $\alpha_j > \alpha_{\min}$, the point y^i might be acceptable if sufficient progress in one of the two measures, relative to the value at the current point x^i, holds:

$$\theta(y^i) \leq (1 - \gamma_\theta)\, \theta(x^i) \;\text{ or }\; f(y^i) \leq f(x^i) - \gamma_f\, \theta(x^i). \tag{6}$$

However, when the current solution is (almost) feasible, in practice, if $\theta(x^i) \leq \theta_{\min}$, the trial point has to satisfy only the condition

$$f(y^i) \leq f(x^i) - \gamma_f\, \theta(x^i) \tag{7}$$

to be acceptable, in order to prevent convergence to feasible but non-optimal solutions, where $0 < \theta_{\min} \ll 1$. In particular, the corresponding solution/iteration

is denoted by f-type. To prevent cycling between points that improve either θ or f, at each iteration t, the algorithm maintains the filter $\mathcal{F}_t$ that contains pairs (θ, f) that are prohibited for a successful trial point at iteration t. Thus, during the line search procedure, a trial point y^i is acceptable only if $(\theta(y^i), f(y^i)) \notin \mathcal{F}_t$.

The filter is initialized to $\mathcal{F}_0 \subseteq \left\{(\theta, f) \in \mathbb{R}^2 : \theta \geq \theta_{\max}\right\}$, where $\theta_{\max} > 0$ is the upper bound on θ, and later is augmented using the formula

$$\mathcal{F}_{t+1} = \mathcal{F}_t \cup \left\{(\theta, f) \in \mathbb{R}^2 : \theta > (1-\gamma_\theta)\theta(x^i) \text{ and } f > f(x^i) - \gamma_f \theta(x^i)\right\}$$

only after every iteration in which the trial point satisfies (6). Since an inclusion of a point into the filter can occur only when $\theta > 0$, an f-type trial solution is never included into the filter.

Restoration Step When it is not possible to find a step size $\alpha_j > \alpha_{\min}$ that satisfy one of the above referred conditions, the algorithm reverts to a restoration step. In this case, an approximate descent direction for θ or f, at x^i, is computed. The procedure that defines the descent direction is the following. Two exploring points e^1, e^2 are randomly generated in a small neighborhood of the point x^i and an approximate descent direction for θ or f is defined [17, 29]. When the current point x^i is feasible, the direction d is descent for f at x^i; otherwise it is descent for θ at x^i. Thus

$$\Delta^j = \begin{cases} f(e^j) - f(x^i), & \text{if } \theta(x^i) \leq \theta_{\min} \\ \theta(e^j) - \theta(x^i), & \text{otherwise} \end{cases}$$

where $\|e^j - x^i\| \leq \varepsilon$ for $j = 1, 2$ and a very small positive constant ε, and

$$d = -\frac{1}{\sum_{k=1}^{2} |\Delta^k|} \sum_{j=1}^{2} \Delta^j \frac{e^j - x^i}{\|e^j - x^i\|}. \tag{8}$$

Based on the direction (8), the trial solution $y^i = x^i + d$ is accepted if it remains inside Ω; otherwise a projection onto Ω is carried out.

4 Numerical Experiments

In this section, the numerical results of a preliminary study running a benchmark set of engineering problems are reported. A comparison with the results available in the literature is also included. The six chosen engineering design problems are the most common in the literature.

– The welded beam design problem has four design variables and seven inequality constraints [6, 17, 29, 34, 40]. The objective is to minimize the cost of a welded beam, subject to the constraints on the shear stress, bending stress, buckling load on the bar, end deflection of the beam and side constraints.

– In the speed reducer design problem [6, 29, 40], the weight of the speed reducer is to be minimized subject to the constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shafts and stress in the shafts. The problem has seven variables and 11 inequality constraints.
– The tension/compression spring design problem has three continuous variables and four constraints [6, 17, 29, 40], and aims to minimize the weight of a tension/compression spring.
– The 3-bar truss design aims to minimize the volume of the truss subject to the stress constraints [6, 29, 40]. This problem has two design variables representing the cross-sectional areas of two bars (two identical of three-bar) and three inequality constraints.
– In the tubular column design, the cost of fabrication is to be minimized [6, 29]. This problem has two design variables with two inequality constraints.
– The cylindrical vessel design [6, 17, 24, 29, 34] (with both ends capped with a hemispherical head) is to minimize the total cost of fabrication. The problem has four design variables (two of them are multiples of 0.0625) and four inequality constraints.

The C++ programming language is used in this real-coded algorithm. The computational tests were performed on a PC with a 2.8 GHz Core Duo Processor P9700 and 6 Gb of memory. The size of the population is defined as $p_{\text{size}} = \min\{50, 5n\}$. Table 1 displays the results of the comparative tests. Since the algorithm relies on some random parameters and variables, we solve each problem 30 times. The best of the solutions found in all 30 runs is denoted in the table by 'sol.', 'S.D.' gives the standard deviation of the obtained objective function values and 'n.f.e.' gives the average number of function evaluations from all the runs. The user defined parameters are set as follows: $\theta_{\max} = 10^4$, $\theta_{\min} = 10^{-6}$, $\gamma_\theta = \gamma_f = 10^{-8}$, $\alpha_{\min} = 10^{-3}$, $\kappa = 0.8$ and $\varepsilon = 10^{-3}$. The parameter ς is not fixed over the iterative process. Initially, is set to one and is reduced until it reaches 0.1. The parameters for the termination criteria are $\epsilon_1 = 10^{-4}$, $\epsilon_2 = 10^{-6}$ and $t_{\max} = 200$.

A comparison with some of the results in the literature follows. Two variants, a modified differential evolution (mDE-r) based on competitive ranking [6] and the differential evolution based on adaptive penalty (DE-AP) [34], are used. The other selected methods for this comparison are: feasibility and dominance rules based on a sufficient reduction of constraint violation or objective function values are implemented with a hybrid electromagnetism-like (HEM) algorithm [29]; a hybrid evolutionary algorithm (HEA) implemented with an adaptive constraint-handling technique [40]; a socio-behavioural (SB) model [2]; an improved harmony search (iHS) proposed by Mahdavi et al. [25]; and finally, a filter method implemented in a simulated annealing algorithm context in [17]. From the preliminary results in Table 1, we may conclude that the performance of the AFSFilter algorithm is comparable to the other ones. For the 3-bar truss and tubular problems, the AFSFilter obtained competitive solutions with reduced function evaluations. The proposed method also converges to a solution of the beam problem that is better than those obtained by SB and HEM, and

the solution obtained for the speed problem is also better than the one obtained by SB. We remark that further research is required in the AFSFilter in order to improve the convergence to the solutions with hight accuracy.

Table 1. Comparative results

Method		Problems					
		Beam	Speed	Spring	3-bar truss	Tubular	Vessel
AFSFilter	sol.	2.382927	2999.151	0.012667	263.8964	26.53351	5946.636†
	S.D.	1.3E-2	2.1E00	7.2E-6	2.5E-3	5.2E-3	5.5E+1
	n.f.e.	38342	65779	23636	5388	9223	45287
SB [2]	sol.	2.4426	3008.08	-	-	-	6171.00
	S.D.	-	-	-	-	-	-
	n.f.e.	19259	19154	-	-	-	12630
mDE-r [6]	sol.	2.380810	2994.320	0.012664	263.8919	26.5311	6059.525
	S.D.	-	-	-	-	-	-
	n.f.e.	30000	35000	15000	10000	10000	30000
FSA [17]	sol.	2.381065	-	0.012665	-	-	5868.765†
	S.D.	-	-	2.2E-8	-	-	2.6E+2
	n.f.e.	56243	-	49531	-	-	108883
iHS [25]	sol.	1.7248‡	-	0.012671	-	-	5849.762†
	S.D.	-	-	-	-	-	-
	n.f.e.	200000	-	30000	-	-	-
HEM [29]	sol.	2.386269	2995.804	0.012667	263.8960	26.53227	6072.232
	S.D.	3.1E-2	1.3E00	8.0E-6	4.9E-5	3.5E-3	5.3E+1
	n.f.e.	28650	51989	9605	17479	25136	20993
DE-AP [34]	sol.	2.38113	-	-	-	-	6059.718
	S.D.	0.0E00	-	-	-	-	0.0E00
	n.f.e.	40000	-	-	-	-	80000
HEA [40]	sol.	2.380957	2994.499	0.012665	263.8958	-	-
	S.D.	1.3E-5	7.0E-2	1.4E-9	4.9E-5	-	-
	n.f.e.	30000	40000	24000	15000	-	-

† all variables are considered continuous; - not available
‡ slightly different problem formulation.

5 Conclusions

In this paper, a filter line search method is implemented in a population-based swarm intelligence algorithm to solve continuous nonlinear constrained global optimization problems. The innovative nature of the work is focused on the integration of the filter methodology, as a constraint-handling technique, into an artificial fish swarm algorithm. Based on a population of current solutions, the AFS algorithm computes trial solutions using search directions and a step size in a way that sufficient progress towards the optimal solution is obtained. Trial solutions are acceptable only if they improve either the constraint violation or

the objective function relative to the current solutions, and are acceptable by the filter.

The preliminary numerical results on six common benchmark engineering design optimization problems show that the proposed AFSFilter method is competitive when compared with other stochastic methods for global optimization, thus encouraging the application of this AFS filter-based paradigm to more complex problems, such as those with discrete variables. We think that the algorithm efficiency can be improved through proper tuning of the algorithm parameters. The impact of some parameters on the performance of the algorithm will be analyzed in the future. An elitist procedure will be implemented in a way that the best point of the population will be maintained regardless being dominated by a trial point at a particular iteration. It has been also observed that accuracy could be improved with an intensification search around the best found solution, at the final stage of the algorithm. This local search aims at improving the final solution at a reduced computational cost and is a common procedure in global optimization methods.

Acknowledgments. The financial support from FEDER COMPETE (Programa Operacional Fatores de Competitividade / Operational Programme Thematic Factors of Competitiveness) and FCT (Fundação para a Ciência e a Tecnologia / Portuguese Foundation for Science and Technology) Project FCOMP-01-0124-FEDER-022674 is gratefully acknowledged.

References

1. Aguirre, A.H., Rionda, S.B., Coello Coello, C.A., Lizárraga, G.L., Montes, E.M.: Handling constraints using multiobjective optimization concepts. International Journal for Numerical Methods in Engineering, 59, 1989–2017 (2004)
2. Akhtar, S., Tai, K., Tay, T.: A socio-behavioural simulation model for engineering design optimization. Engineering Optimization, 34, 341–354 (2002)
3. Ali, M.M., Golalikhani, M.: An electromagnetism-like method for nonlinearly constrained global optimization. Computers and Mathematics with Applications, 60, 2279–2285 (2010)
4. Audet, C., Dennis, Jr., J.E.: A pattern search filter method for nonlinear programming without derivatives. SIAM Journal on Optimization, 14(4) 980–1010 (2004)
5. Azad, M.A.K., Fernandes, E.M.G.P., Rocha, A.M.A.C.: Nonlinear continuous global optimization by modified differential evolution. International Conference of Numerical Analysis and Applied Mathematics 2010, T.E Simos et al. (Eds.) 1281, 955–958 (2010)
6. Azad, M.A.K., Fernandes, E.M.G.P.: Modified differential evolution based on global competitive ranking for engineering design optimization problems. Lecture Notes in Computer Science 6784, B. Murgante et al. (Eds.) 245–260 (2011)
7. Barbosa, H.J.C., Lemonge, A.C.C.: An adaptive penalty method for genetic algorithms in constrained optimization problems. Frontiers in Evolutionary Robotics, H. Iba (Ed.) 9–34, I-Tech Education Publ., Austria (2008)
8. Birgin, E.G., Floudas, C.A., Martinez, J.M.: Global minimization using an augmented Lagrangian method with variable lower-level constraints. Mathematical Programming, 125, 139-162 (2010)

9. Chootinan, P., Chen, A.: Constrained handling in genetic algorithms using a gradient-based repair method. Computers and Operations Research, 33, 2263–2281 (2006)
10. Coello Coello, C.A.: Use of a self-adaptive penalty approach for engineering optimization problems. Computers in Industry, 41, 113–127 (2000)
11. Costa, M.F.P., Fernandes, E.M.G.P.: Assessing the potential of interior point barrier filter line search methods: nonmonotone versus monotone approach, Optimization, 60(10-11) 1251–1268 (2011)
12. Costa, M.F.P., Fernandes, E.M.G.P.: On minimizing objective and KKT error in a filter line search strategy for an interior point method, Lecture Notes in Computer Science 6784, B. Murgante et al. (Eds.) 231–244 (2011)
13. Deb, K.: An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering, 186, 311–338 (2000)
14. Fernandes, E.M.G.P., Martins, T.F.M.C., Rocha, A.M.A.C.: Fish swarm intelligent algorithm for bound constrained global optimization. CMMSE 2009, J.V. Aguiar (Ed.) 461–472 (2009)
15. Fletcher, R., Leyffer, S.: Nonlinear programming without a penalty function. Mathematical Programming, 91, 239–269 (2002)
16. Hedar, A.-R., Fukushima, M.: Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization. Optimization Methods and Software, 19, 291–308 (2004)
17. Hedar, A.-R., Fukushima, M.: Derivative-free filter simulated annealing method for constrained continuous global optimization. Journal of Global Optimization, 35, 521–549 (2006)
18. Gao, X.Z., Wu, Y., Zenger, K., Huang, X.: A knowledge-based artificial fish-swarm algorithm. 13th IEEE International Conference on Computational Science and Engineering, 327–332 (2010)
19. Jiang, M., Mastorakis, N., Yuan, D., Lagunas, M. A.: Image segmentation with improved artificial fish swarm algorithm. ECC 2008, Lecture Notes in Electrical Engineering 28, N. Mastorakis et al. (Eds.) 133–138 (2009)
20. Jiang, M., Wang, Y., Pfletschinger,S., Lagunas, M. A., Yuan, D.: Optimal multiuser detection with artificial fish swarm algorithm. International Conference on Intelligent Computing 2007, CCIS 2, D.-S. Huang et al. (Eds.), 1084–1093 (2007)
21. Kaelo, P., Ali, M.M.: A numerical study of some modified differencial evolution algorithms. European Journal of Operational Research, 169, 1176–1184 (2006)
22. Karaboga, D., Basturk, B.: Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems. Foundations of Fuzzy Logic and Soft Computing, Lecture Notes in Computer Science 4529, P. Melin et al. (Eds.) 789–798 (2007)
23. Karimi, A., Nobahari, H., Siarry, P.: Continuous ant colony system and tabu search algorithms hybridized for global minimization of continuous multi-minima functions. Computational Optimization and Applications, 45, 639–661 (2010)
24. Liu, J.-L., Lin, J.-H.: (2007) Evolutionary computation of unconstrained and constrained problems using a novel momentum-type particle swarm optimization. Engineering Optimization, 39, 287–305 (2007)
25. Mahdavi, M., Fesanghary, M., Damangir, E.: An improved harmony search algorithm for solving optimization problems. Applied Mathematics and Computation, 188, 1567–1579 (2007)
26. Mallipeddi, R., Suganthan, P.N.: Ensemble of constraint handling techniques. IEEE Transactions on Evolutionary Computation, 14, 561–579 (2010)

27. Petalas, Y.G., Parsopoulos, K.E., Vrahatis, M.N.: Memetic particle swarm optimization. Annals of Operations Research, 156, 99–127 (2007)
28. Pereira, A.I., Costa, M.F.P., Fernandes, E.M.G.P.: Interior point filter method for semi-infinite programming problems, Optimization, 60(10-11) 1309–1338 (2011)
29. Rocha, A.M.A.C., Fernandes, E.M.G.P.: Hybridizing the electromagnetism-like algorithm with descent search for solving engineering design problems. International Journal of Computer Mathematics, 86, 1932-1946 (2009)
30. Rocha, A.M.A.C., Fernandes, E.M.G.P., Martins, T.F.M.C.: Novel fish swarm heuristics for bound constrained global optimzation problems. ICCSA 2011, Part III, Lecture Notes in Computer Science 6784, B. Murgante et al. (Eds.) 185–199 (2011)
31. Rocha, A.M.A.C., Fernandes, E.M.G.P.: Numerical study of augmented Lagrangian algorithms for constrained global optimization. Optimization, 60(10-11) 1359–1378 (2011)
32. Rocha, A.M.A.C., Martins, T.F.M.C., Fernandes, E.M.G.P.: An augmented Lagrangian fish swarm based method for global optimization. Journal of Computational and Applied Mathematics, 235(16) 4611-4620 (2011)
33. Runarsson, T.P., Yao, X.: Stochastic ranking for constrained evolutionary optimization. IEEE Transaction on Evolutionary Computation, 4, 284–294 (2000)
34. Silva, E.K., Barbosa, H.J.C., Lemonge, A.C.C.: An adaptive constraint handling technique for differential evolution with dynamic use of variants in engineering optimization. Optimization and Engineering, 12, 31–54 (2011)
35. Socha, K., Dorigo, M.: Ant colony optimization for continuous domains. European Journal of Operational Research, 185, 1155–1173 (2008)
36. Stanoyevitch, A.: Homogeneous genetic algorithms. International Journal of Computer Mathematics, 87, 476–490 (2010)
37. Ulbrich, M., Ulbrich, S., Vicente, L.N.: A globally convergent primal-dual interior-point filter method for nonlinear programming. Mathematical Programming, 100, 379–410 (2004)
38. Wächter, A., Biegler, L.T.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming, 106, 25–57 (2006).
39. Wang, C.-R., Zhou, C.-L., Ma, J.-W.: An improved artificial fish-swarm algorithm and its application in feed-forward neural networks. Proceedings of the 4th ICMLC, 2890–2894 (2005)
40. Wang, Y., Cai, Z., Zhou, Y., Fan, Z.: Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique. Structural and Multidisciplinary Optimization, 37(4) 395–413 (2009)
41. Wang, X., Gao, N. , S. Cai, Huang, M.: An artificial fish swarm algorithm based and ABC supported QoS unicast routing scheme in NGI. ISPA 2006, Lecture Notes in Computer Science 4331, G. Min et al. (eds.) 205–214. Springer-Verlag (2006)
42. Zahara, E., Hu, C.-H.: Solving constrained optimization problems with hybrid particle swarm optimization, Engineering Optimization, 40(11) 1031–1049 (2008)